Microsoft SQL Server – Dev/Programmer Configuration Guide

TL;DR – Don’t have the time right now to read this entire document? - Jump to the end for a quick summary.

Purpose:

As a continuation of the article “You don’t need to be a sysadmin” - by Kenneth Fisher. This document describes a method to provide access to developers / programmers and related support personal (“Devs”) who maybe responsible to deliver solutions to customers in a highly controlled database server environment.

Scope:

This document is specific to configuration of SQL Server 2012 and above. While some of the principles maybe shared with other DBMS. It does not deal with other databases such as Oracle, MySQL
etc.

Specifically this document describes

a) Creation of a special server role that can be applied that gives troubleshooting / configuration access without requiring sysadmin privileges.

b) The ability to allow Devs to create databases, and associate logins, manage passwords, and remove logins **without the sysadmin role** and without granting the ability to manage all logins on the entire instance.

For this guideline we will be dealing with a fictional user “GeekWisdom” who will be granted a specific secured account (gwSecure). This server role we will create will be called “DevProgrammer”, and we will use the password **MyPassword.** We will be making use of a second account called “saproxy”. These names can be configured as need for your particular environment.

Warning: Instructions are provided for NON-PRODUCTION environments in situations where sysadmin is not available to Devs.

Part 1: DevProgrammer SERVER Role

It is sometimes thought that simply giving ‘db_owner’ on related databases is sufficient for most daily functions preformed by a programmer. However, there exists a very important business case that can sometimes be forgotten. Programmers, by their very nature, are natural problems solvers - creative people, who love to explore and learn. A person who is excited about developing, excited about the next challenge, solving that next problem will be far more effective to an organization when provided with an environment that supports that creativity, out of the box thinking, and problem solving.

We have a word in our lingo for that environment it’s called “DEV”, other words include “SCRATCH”, “PLAYGROUND” or “SANDBOX” . It is our world, our safe space – our home. Try to take away the

Devs tools to safely explore and you risk compromising the very reason they were hired in the first place.

Fortunately, we can solve this quite easily in an SQL Server environment. We can create a role specifically for that purpose.

```
USE [master]
GO
CREATE SERVER ROLE [DevProgrammer]

GRANT CONNECT ON ENDPOINT::[Dedicated Admin Connection] TO [DevProgrammer]
GRANT CREATE ANY DATABASE TO [DevProgrammer]
GRANT VIEW DEFINITION ON ENDPOINT::[Dedicated Admin Connection] TO [DevProgrammer]
GRANT VIEW DEFINITION ON LOGIN::[sa] TO [DevProgrammer]
GRANT CONNECT SQL TO [DevProgrammer]
GRANT VIEW ANY DATABASE TO [DevProgrammer]
GRANT VIEW ANY DEFINITION TO [DevProgrammer]
GRANT VIEW SERVER STATE TO [DevProgrammer]
GO
```

Figure 1: The DevProgrammer SERVER ROLE

These simple steps, allow the DevProgrammer to explore all aspects of the SQL Server in a safe way. They can create databases, view existing databases, view all definitions and server state. This allows them to understand the "why" when something does not seem to work in the Devs environment, but at the same time it does not allow them to change specific settings which could possibly introduce a security risk.

Note: A Dev or "Scatch" environment, should never contain confidential data, it is a playground to try new things out to see if a solution will meet a specific business need.

This role let's the DEV be a DEV! - Now they can prototype a specific solution or simply create a proof of concept. It can be used to test the impact of a server-wide setting change on an existing application, or troubleshoot while something that worked yesterday suddenly seems to have stopped working today. This exploration allows the Dev to learn, grow, and share new found knowledge with fellow collegues. In fact, the only reason I am able to share this particular document is due to the great respect I have had for jobs in the past which respected this very principle of exploration, and creativity.

Some Dev's such as GeekWisdom will seek methods to solve problems, and without the necessary environment may resort to choices such as Microsoft Access, Excel which may not provide sufficient capability resulting in BIG problems for the client later down the line.

Let avoid this and work with our our friendly developer ‘GeekWisdom’ access to this new Server Role

```
USE [master]
ALTER SERVER ROLE [DevProgrammer] ADD MEMBER [GWSECURE]
GO
```

Figure 2 – Granting GeekWisdom our new DevProgrammer Server role

Part 2: LOGIN PERMISSIONS

Now that we have proudly worked together - Unfortunately, all is not perfect in GeekWisdom’s life. We have granted him the ability to create a database for his next project. Unfortunately, he cannot create a login on the server that connects to his new database 😔

But we have a problem. We can’t simply grant him access to ALTER ANY LOGIN because this potentially could lead to privilege elevation.

At first glance, one might think ‘no big deal’. He can simply contact a DBA to setup a login for him. However this leads to a few important problems

a) DBA’s maybe busy, the DEV simply needs to test something quicky for a tight turn-around that could be delayed.

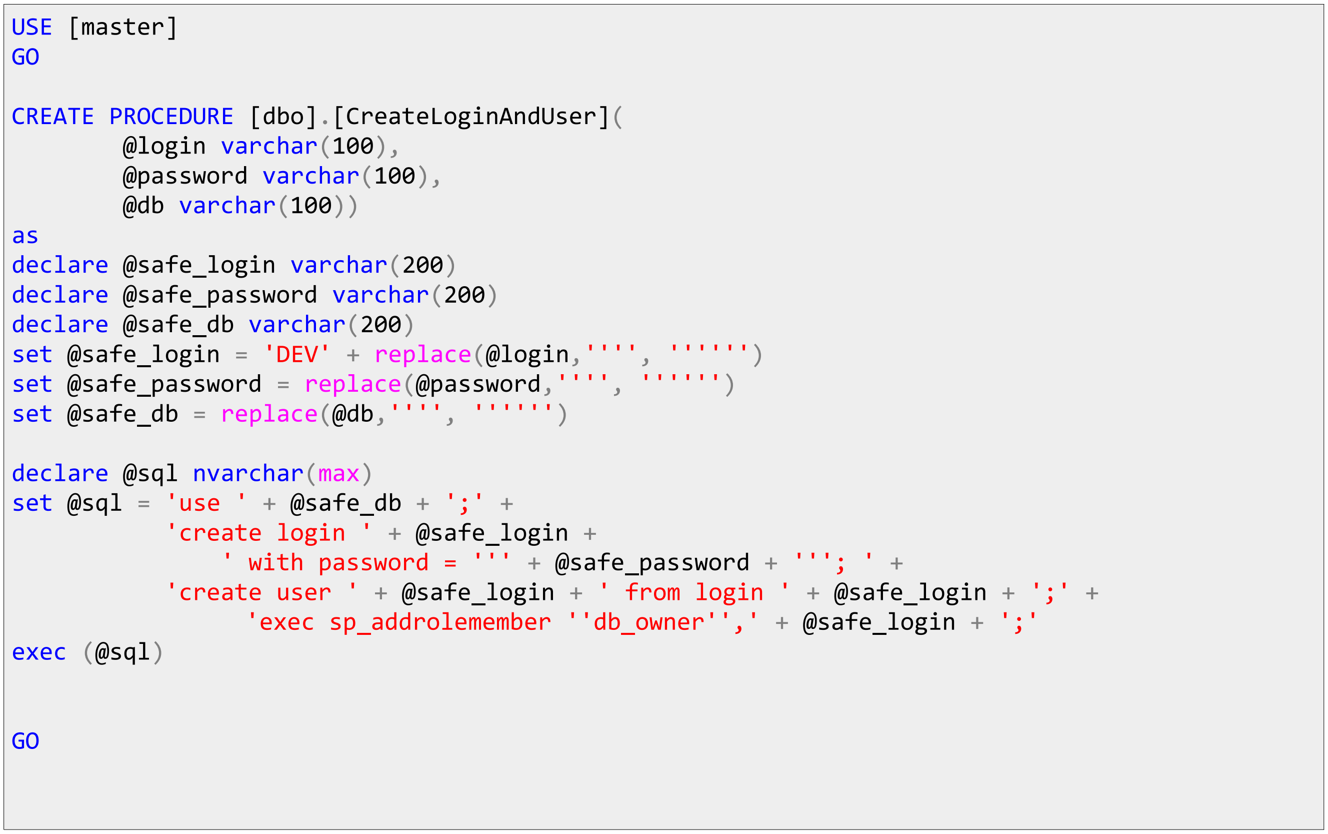
b) The DBA has to prioritize their own work. Taking time out of their schedule could impact others whom they serve, and would not be an efficient use of resources

c) If under pressure, the DEV may choose to simply map his current login to the new database, and put his current login credentials in configuration files. These credentials unfortunately if stolen could then be used to gain access to many more resources than just the Development environment. Especially dangerous if using Windows credentials that secure other files, servers, email. In this situation we created a much worse security problem then we had hoped to solve by denying the ability to create logins on the same server!

Fortunately we can solve this problem with the use of some stored procedures. By granting our friendly GeekWisdom only access to these stored procedures (and not ALTER LOGIN) permission. The ability to create, delete, and manage logins without the ability of SYSADM or PRIVILEGED ELEVATION.

Stored Procedure Call	Purpose
CreateLoginAndUser(@login, @password, @db) OoONB	Creates a new login with username @login and password @password, add’s the user @login to the DB @db as db_owner. The login is added with the word DEV appended to the login name to distinguish it from logins not created by the stored procedure (ie: someone with actual sysadm ability)
AlterUserPassword(@login,@newpassword,@oldpassword)	Change the password of @login. Note: The user MUST know the existing password for this to work and it MUST be one of

	the DEV logins (ie: created from the CreateLoginAndUser above)
DropLoginAndUser(@login, @db)	Removes the Login from the server and removes the user from the DB (if possible)



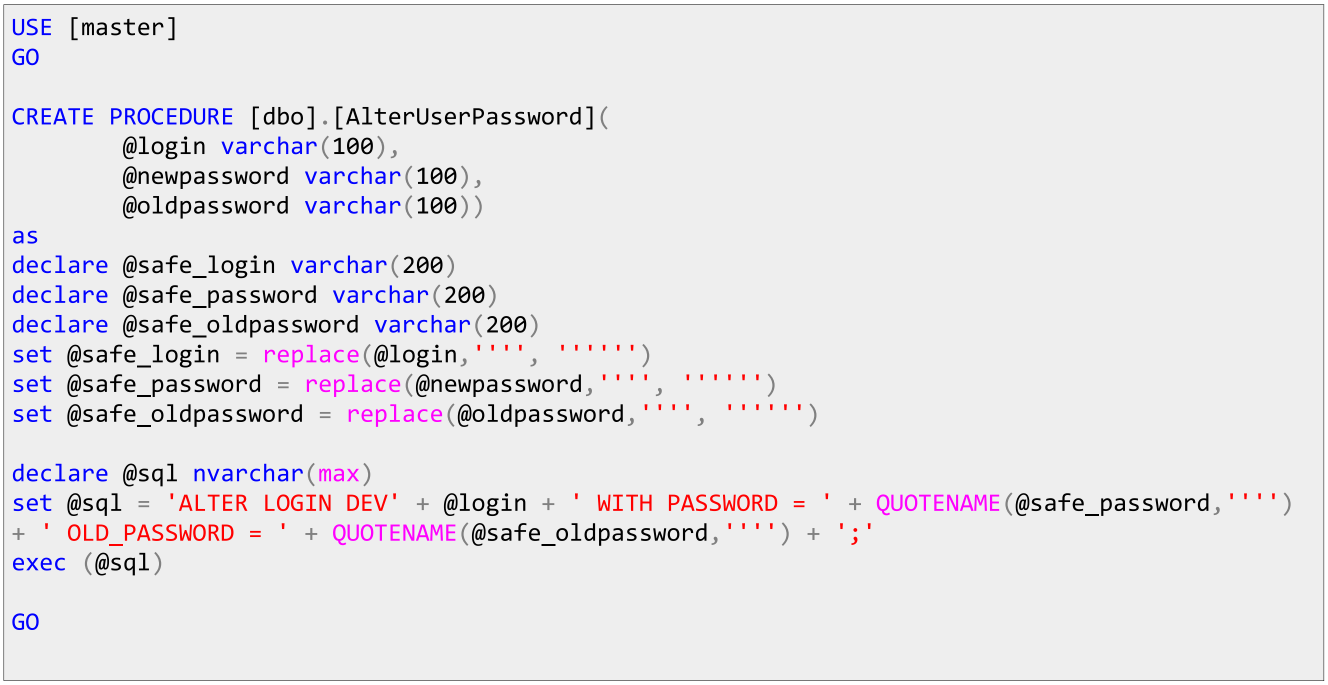
```
USE [master]
GO

CREATE PROCEDURE [dbo].[CreateLoginAndUser](
        @login varchar(100),
        @password varchar(100),
        @db varchar(100))
as
declare @safe_login varchar(200)
declare @safe_password varchar(200)
declare @safe_db varchar(200)
set @safe_login = 'DEV' + replace(@login,'''', '''''')
set @safe_password = replace(@password,'''', '''''')
set @safe_db = replace(@db,'''', '''''')

declare @sql nvarchar(max)
set @sql = 'use ' + @safe_db + ';' +
           'create login ' + @safe_login +
               ' with password = ''' + @safe_password + '''; ' +
           'create user ' + @safe_login + ' from login ' + @safe_login + ';' +
                 'exec sp_addrolemember ''db_owner'',' + @safe_login + ';'
exec (@sql)


GO
```

Figure 3: The new ‘CreateLoginAndUser’



```
USE [master]
GO

CREATE PROCEDURE [dbo].[AlterUserPassword](
        @login varchar(100),
        @newpassword varchar(100),
        @oldpassword varchar(100))
as
declare @safe_login varchar(200)
declare @safe_password varchar(200)
declare @safe_oldpassword varchar(200)
set @safe_login = replace(@login,'''', '''''')
set @safe_password = replace(@newpassword,'''', '''''')
set @safe_oldpassword = replace(@oldpassword,'''', '''''')

declare @sql nvarchar(max)
set @sql = 'ALTER LOGIN DEV' + @login + ' WITH PASSWORD = ' + QUOTENAME(@safe_password,'''')
+ ' OLD_PASSWORD = ' + QUOTENAME(@safe_oldpassword,'''') + ';'
exec (@sql)

GO
```

Figure 4: The new ‘AlterUserPassword’

```
ALTER PROCEDURE [dbo].[DropLoginAndUser](
        @login varchar(100),
        @db varchar(100))
as
declare @safe_login varchar(200)
declare @safe_db varchar(200)
set @safe_login = 'DEV' + replace(@login,'''', '''''')
set @safe_db = replace(@db,'''', '''''')

declare @sql nvarchar(max)
set @sql = 'use ' + @safe_db + ';' +
           'drop user [' + @safe_login + '];'  +
           'drop login [' + @safe_login +  '];'


exec (@sql)
```

Figure 3: The new ‘DropLoginAndUser’

Let’s grant access to these new stored procedures to our DEV GeekWisdom

```
USE MASTER
GO
GRANT EXECUTE ON [dbo].[CreateLoginAndUser] TO [GWSECURE]
GO
GRANT EXECUTE ON [dbo].[AlterUserPassword] TO [GWSECURE]
GO
GRANT EXECUTE ON [dbo].[DropLoginAndUser] TO [GWSECURE]
GO
```

Now, unfortunately, we still do have a problem. While stored procedures can be a great way to provide underlying access to information. Generally speaking the procedures above will fail when executed by GWSECURE because they will run AS GWSECURE which does not have CREATE LOGIN or ALTER LOGIN permissions.

Fortunately, there is a way to have stored procedures NOT run as GWSECURE but instead run as a user with SYSADM using digital signatures.

1) First we need to create a certificate (must be done as a user with SYSADM)

```
CREATE CERTIFICATE [saproxyCert]
   ENCRYPTION BY PASSWORD = N'MyPassword'
   WITH SUBJECT = N'StoredProceduresProxy';
```

2) Next we create a user with that cert and give the user sysadmin

```
CREATE LOGIN [saproxy] FROM CERTIFICATE [saproxyCert];
GO
ALTER SERVER ROLE [sysadmin] ADD MEMBER [saproxy];
```

3) Finally we digitally sign the stored procesures with saproxy’s signatures

```
ADD SIGNATURE
    TO [dbo].[DropLoginAndUser]
    BY CERTIFICATE [SAProxyCert]
    WITH PASSWORD = 'MyPassword';

ADD SIGNATURE
    TO [dbo].[CreateLoginAndUser]
    BY CERTIFICATE [SAProxyCert]
    WITH PASSWORD = 'MyPassword';

ADD SIGNATURE
    TO [dbo].[AlterUserPassword]
    BY CERTIFICATE [SAProxyCert]
    WITH PASSWORD = 'MyPassword';

ADD SIGNATURE
    TO [dbo].[DropLoginAndUser]
    BY CERTIFICATE [SAProxyCert]
    WITH PASSWORD = 'MyPassword';
```

And viola! - Now when GeekWisdom tries to execute these stored procedures they will run under the ‘saproxy’ username. GeekWisdom will not be able to change these stored procedures and if he/she tried to create ‘copies’ they would not run without the digital singature.

Part 3: SQL Jobs

Regular recurring jobs or ‘BATCHES”, are a common solution to various business problems. A nightly batch might sync new records for multiple sources or ‘turn on/off’ a key piece of the system at a specified time each night. From a development perspective, the DEV might need to verify something in a schedule job first, or might be migrating the job from one machine to another and needs to run things concurrently to ensure no loss of data and that the batch is working properly.

Sometimes an existing batch might need turned off quickly due to a configuration error which may put data at risk if it runs and must be urgently stopped.

So let's grant our GeekWisdom buddy the ability to manage scheduled jobs (create, add, delete)

```
USE MSDB;
GO
CREATE USER [GWSECURE] FOR LOGIN [GWSECURE] WITH DEFAULT_SCHEMA=[dbo]
EXEC sp_addrolemember 'SQLAgentOperatorRole', 'GWSECURE';
```

Summary of Points:

- The 'sysadmin' role is a highly privileged role that grants all permissions to a user, and must be both given and used with **extreme caution in environments where the security of underlying data is of upmost importance**. Even where environments are less strict it is important to balance both the needs of the Developer where this may differ from the needs of those responsible and accountable to supporting that environment.

- Several business cases exist for the need of Developers to troubleshoot problems, verify configuration, create / update databases. Specifically these inlcude; solution design, proof of concept of solution, prototyping. This access promotes both creativity and problem solving while providing no requirement for sysadmin access. This can be preformed with the application of a custom Server Role.

- The absence of balance of needs can inadvertently lead to more serious security risks. For example, the use of the Dev's Windows credentials in order to provide a "quick" proof of concept could prove fatal to the overall security of the company must be avoided at all costs. The Dev attempting to use other tools such as Microsoft Access or Excel in the absence of access to SQL Server may prove a viable solution at first, but risks problems with the long term viability and capability. This can be avoided with the addition application of signed stored procedures providing the necessary need to both create and assign specific logins fit only for specific purposes, without requiring the need to grant the ability to elevate the Dev's privilege to what otherwise would only be available to a sysadmin

- Things will go wrong, and mistakes will happen – such is the nature of being human. Effectively and Efficient management requires the working together of all involved parties even when things go wrong. While accountability is an important consideration, when things go wrong teamwork is frequently essential to the problem solving effort. One specific case is that of a rogue batch job which may fail. DBA teams maybe able to identify the failure, while the Dev may be required to understand the impact, and a lead required to communicate that lead to back the business. Everyone has roles to preform. By providing access to create and control those jobs, the Dev can both effectively and efficiently provide explanation and solve problems early in the process before a small glitch has the opportunity to becomes a major fire.